ENVPREDRSCHFAC COMPUTER PROGRAMMING NOTE 13
MARCH 1974

2

USERS GUIDE TO UPDATE

BY

JAMES LONG

Naval 407279

ENVIRONMENTAL PREDICTION RESEARCH FACILITY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

ENVPREDRSCHFAC
Computer Programming Note No.13

6

USERS GUIDE TO UPDATE,

by

10 JAMES/LONG

12 20p.

11 MARCH 1974

14 NEPRF-CP-Note-13

ENVIRONMENTAL PREDICTION RESEARCH FACILITY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

407279

1B

CONTENTS

FOREWORD

UPDATE is a system which makes it possible to modify programs without handling the entire source deck. UPDATE works on both FTN and COMPASS decks, however, the examples presented here are exclusively FTN. This programming note is designed to provide the CDC 6500 programmer with sufficient insight to effectively use the UPDATE system.

ACKNOWLEDGEMENTS

Grateful appreciation is extended to Mr. Robert E. Walker, Head, Numerical Applications Department, ENVPREDRSCHFAC and CDR H.D. Hamilton, Data Systems Officer, FLENUMWEACEN for their technical assistance in preparing this document.

1. GENERAL DESCRIPTION

A program deck is submitted to UPDATE prefaced by a *DECK card which specifies the name by which the deck will be known to UPDATE. The deck may be divided into several sections with different names by using several *DECK cards. UPDATE, called with the N parameter, makes a PROGRAM LIBRARY (PL) from this deck. The PL contains a modified copy of the original deck, where each card has been assigned an identifying name and sequence number. The name is the one that appeared on the last *DECK card.

Later, when some changes are to be made to the program, a small deck of UPDATE directives is assembled indicating the modifications to be made. This deck, called a CORRECTION SET, specifies where the additions and/or deletions are to be made by referring to the identifying names and sequence numbers on the original PL. Some directives used in CORRECTION SETS are described later in this writeup (section III).

To run the program with the modifications, the CORRECTION SET is submitted to UPDATE, which updates the original program library (known in this context as the OLDPL) and puts the complete version (in a format suitable for the compiler) onto a file called COMPILE which may then be compiled and executed. The OLDPL, however, <u>is not permanently changed</u>. The CORRECTION SET should be resubmitted with each run, until a permanent new program library (NEWPL) is created.

When the deck of corrections has reached a certain size or complexity, it is possible to have the changes permanently incorporated into the program library. This is done by calling UPDATE, as in the initial creation run, with the N option. This creates a new program library on the file NEWPL. This NEWPL has incorporated <u>all</u> the changes indicated by the CORRECTION SET(s).

Two other features of UPDATE are particularly useful. First, along with the particular program, blocks of cards called COMDECKS (because their most obvious use is for FORTRAN common block declarations) may be defined. These are kept separate from the rest of the program and subroutines, but may be inserted as desired. So, changing a common block once, changes it in every subroutine which uses it.

Second, if given the proper command, UPDATE will serve up only the subroutines changed. In the case of a standard program, in which most of the subroutines are never changed, a whole program may be compiled once and the compiled version may be stored along with the card images. Thereafter, only a few subroutines are changed and only those changed are compiled. Then, with COPYL, the pre-compiled version is copied with the new version to form a new program. COPYL recognizes the fact that it already has the revised routines, so it skips the old versions of these routines. The result -- a super program that used to take three minutes just to compile can now be run, or at least debugged, as an "instant" job.

This programming note is by no means a complete discussion of UPDATE. For a description of the many facets of UPDATE not presented here, see the UPDATE REFERENCE MANUAL.

2. EXAMPLES OF USAGE

In all of the examples presented here a fictional programmer named HERO is used, who runs programs under job code FREE. Programmers must substitute their own name and job code wherever Hero and his job code appear. The W option is used to force a sequential PL even when the PL goes to tape (see UPDATE REF MAN). The FTN card always uses the A option, (skip to EXIT(S) if compilation errors occur). Lastly, all tapes are UNLOADED or RETURNED as soon as possible. If the EXIT(S) control card does not exist the job simply aborts, if the EXIT(S) card does exist control skips to this card and all control cards after EXIT(S) are executed.

2.1 GENERATION OF AN UPDATE PROGRAM LIBRARY ON PUNCH CARDS

Purpose: To "compress" the size of the deck submitted to the 6500 card reader.

2.1.1 Sample Card Deck For "Creation Run"

```
FREE1,T20,S7.HERO
UPDATE(N=PUNCHB,W)
FTN(I=COMPILE,A)
LGO.
EXIT(S)
7/8/9
*DECK EMBRYO
   PROGRAM EMBRYO (INPUT, OUTPUT)
   END
*DECK BIRDS
   SUBROUTINE BIRDS (I,J), RETURNS (N,M)
   END
*DECK BEES
   FUNCTION BEES (K)
   END
7/8/9
   DATA
6/7/8/9
```

The program library (NEWPL) is written on the PUNCHB file and punched. The FTN lists the cards and their associated UPDATE indentifiers.

2.1.2 Running From Punched Output

```
FREE2,T30,S7.HERO
COPYBR(INPUT,OLDPL)
UPDATE(F,W)
FTN(I=COMPILE,A)
LGO.
EXIT(S)
7/8/9
```

```
    UPDATE DECK (PRODUCED IN 2.1.1)
7/8/9
    CORRECTION SET
7/8/9
    DATA
6/7/8/9
```

NOTE: if no corrections are required, simply use TWO 7/8/9 cards (i.e., a blank record). In other words, keep the same number of 7/8/9 cards present in the deck.

2.2 GENERATION OF AN UPDATE PROGRAM LIBRARY ON TAPE

Purpose: To allow the programmer to submit a deck containing only the corrections and DATA cards, as the program itself is on tape.

2.2.1 <u>Sample Card Deck For Generation Of UPDATE PL On Tape</u>

```
FREE3,T34,TU1,S7.HERO
REQUEST NEWPL,HI,VSN=74321.  RING IN
UPDATE(N,W,F)
UNLOAD(NEWPL)
FTN(I=COMPILE,A)
REQUEST TAPE1,HY,VSN=75402.  RING IN 800 BPI
LGO.
EXIT(S)
7/8/9
    (SOURCE DECK AND ASSOCIATED *DECK CARDS
    AS IN 2.1.1)
7/8/9
    DATA
6/7/8/9
```

2.2.2 Sample Card Deck For Creation Of NEWPL From Cards In 2.1.2

```
FREE4,TU1,T45,S7.HERO
COPYBR(INPUT,OLDPL)
REQUEST NEWPL,HI,VSN=74321.  RING IN
UPDATE(F,N,W)
UNLOAD(NEWPL)
FTN(I=COMPILE,A)
REQUEST TAPE1,HY,VSN=75402.  RING IN 800 BPI
LGO.
EXIT(S)
7/8/9
   UPDATE DECK (produced in 2.1.1)
7/8/9
   UPDATE CORRECTION SET
7/8/9
   DATA
6/7/8/9
```

NOTE: In either case (2.2.1 or 2.2.2) a NEWPL was created on VSN 74321.

2.2.3 Sample Card Deck To RUN From VSN 74321

```
FREES,TU1,T100,S7.HERO
REQUEST OLDPL,VSN=74321.  RING OUT
UPDATE(F,W)
UNLOAD(OLDPL)
FTN(I=COMPILE,A)
REQUEST TAPE1,HI,VSN=75000.RING IN
LGO.
EXIT(S)
7/8/9
```

```
         UPDATE CORRECTION SET OR BLANK RECORD
         (just the TWO 7/8/9's)
      7/8/9
         DATA
      6/7/8/9
```

2.3 USING COMDECKS

The following sample card deck may be used for setting up a COMDECK/UPDATE tape.

```
      FREEJ,T20,TU1,LL10,S7.HERO
      REQUEST NEWPL,VSN=1234.RING IN
      UPDATE(N,W)
      RETURN(NEWPL)
      FTN(I=COMPILE)
      7/8/9 CARD
      *COMDECK HOOPER
         COMMON/HOOPER/X, XY, XYZ (17)
      C
      *COMDECK WHOOPS
         COMMON/WHOOPS/ERROR, MESSAGE (20)
      C
      *DECK GROAN
         PROGRAM GROAN (INPUT, OUTPUT)
      *CALL HOOPER
      *CALL WHOOPS
         DATA XY /1.0/
         .  ETC.
         STOP
         END
      *DECK SUB 1
      *CALL HOOPER
```

```
*CALL WHOOPS
   X=XYZ (7)
   .  ETC.
   RETURN
   END
7-8-9
DATA Cards
6-7-8-9
```

NOTE 1: The common blocks /HOOPER/ and /WHOOPS/ are set up as a separate part of the PL and are declared in more than one subroutine. The *CALL cards are maintained on the PL like any other card, but when UPDATE is making a compile file and encounters a *CALL card, it replaces the card with all the cards in the corresponding COMDECK.

NOTE 2: Each COMDECK ends with a COMMENT card. This isn't necessary, but it is handy. The last card of the COMDECK never has to be deleted; and the identifier of the last card always appears in the listing of active cards.

2.4 USING ADDFILE

The following card deck may be used for the addition of new decks.

```
FREE7,TU2,T700,S7.HERO
REQUEST OLDPL,VSN=74321.  RING OUT
REQUEST NEWPL,VSN=74322.  RING IN.
UPDATE(F,N,W)
RETURN(OLDPL,NEWPL)
FTN(I=COMPILE,A)
LGO.
EXIT(S)
7/8/9
```

```
     UPDATE CORRECTION SET (IF ANY)
*ADDFILE INPUT
*DECK SINCH
     SUBROUTINE SINCH
     END
*DECK COSH
     SUBROUTINE COSH
     END
7/8/9
     DATA
6/7/8/9
```

With the above ADDFILE directive the programmer can insert new decks in his update PL stream. The ADDFILE directive allows one to place the DECKS anywhere in the PL. *ADDFILE, INPUT places them at the end of the current OLDPL.

2.5 BEAUTIFICATION OF THE LISTING

A point may be reached where the entire program consists of corrections, and the identifiers in columns 73-90 are a jumbled mass of many different names. This situation is confusing and makes it difficult to locate specific cards. The solution is to re-sequence the PL. The result is a new PL with completely renamed cards. As in the initial creation run, the names are taken from the *DECK and *COMDECK cards. The NEWPL produced will have all inactive cards purged. The following sample card deck may be used:

```
FREET,TU2,LL20,T1000, S7.  HERO
REQUEST.OLDPL,VSN=74322.  RING OUT
REQUEST NEWPL,VSN 74323.  RING IN
UPDATE(F,N,W)
RETURN(OLDPL,NEWPL)
FTN(I=COMPILE,A)
```

```
LGO.
EXIT(S)
7/8/9
   UPDATE CORRECTION SET
   *SEQUENCE FIRST DECK NAME.LAST DECK NAME
7/8/9
   DATA
6/7/8/9
```

2.6 COPYL AND UPDATE

To create a 2 File (UPDATE/OBJECT CODE) Tape for COPYL use, the following sample card deck may be used:

```
FREE9,T1000,TU1,S7.  HERO
REQUEST OLDPL,VSN=74323.  RING OUT
UPDATE(F,N,W)
UNLOAD(OLDPL)
FTN(I=COMPILE,A)
REWIND(LGO)
REQUEST PROGTAP,VSN=74321.  RING IN.
COPYBF(NEWPL,PROGTAP)
COPYBF(LGO,PROGTAP)
EXIT(S)
7/8/9
   UPDATE CORRECTION SET
6/7/8/9
```

Should there be an error in compilation, VSN 74321 would not be written on due to the A parameter. Tape VSN 74321 now has two files. File #1 = UPDATE PL, File #2 = object CODE (BINARY DECK) of all routines in the PL.

2.7 UPDATE(Q) - THE SELECTIVE UPDATE

Three example card decks are given.

2.7.1 Sample Card Deck for Simple Quick UPDATE

```
FREEQ,TU1,T700,S7.  HERO
REQUEST OLDPL,VSN=74321.  RING OUT.
UPDATE(Q)
UNLOAD(OLDPL)
FTN(I=COMPILE,L=0,A)
REQUEST TAPE1, VSN=..............   .
LGO.
EXIT(S)
7/8/9
  UPDATE CORRECTION SET
*COMPILE EMBRYO,BIRDS,SINH,JOE,SAM,ELMO
7/8/9
DATA
6/7/8/9
```

NOTE: *Only those deck names on the *COMPILE card are written on the COMPILE file. In effect, VSN=74321 is a Library Tape.

2.7.2 Sample Card Deck for Quick UPDATE and Source Deck "Merging". (This example uses an FTN source deck plus the PL.)

```
FREEU,TU1,T700,S7.  HERO
FTN(A)  Note:  this compiles and list source deck)
REQUEST OLDPL,VSN=74321.  RING OUT.
UPDATE(Q)
UNLOAD(OLDPL)
FTN(I=COMPILE,A) Note:  This compiles DECKS on
                        *COMPILE Card
REQUEST TAPE7,VSN=12584.  RING IN.
```

```
LGO.
EXIT(S)
7/8/9
    FTN SOURCE DECK
7/8/9
    CORRECTION SET
*COMPILE FUNC1,FUNC2
7/8/9
    DATA CARDS
6/7/8/9
```

NOTE: The first FTN card compiles the source deck and the second compiles FUNC1 and FUNC2. Thus, the OLDPL has been used like a true library tape.

2.7.3 <u>UPDATE and COPYL Usage</u>

This example is especially useful when only a few minor changes to a program are required and the rest of it is compiled as before. With the use of UPDATE, COPYBF, COPYL directives, and the TAPE created in 2.6, the whole job may be run while compiling only those decks which have been changed and loading a precompiled version of the rest of the decks.

```
FREEZ,TU1,T100,S7.
REQUEST TAPE2,VSN=74321.  RING OUT
COPYBF(TAPE2,OLDPL)
COPYBF(TAPE2,PROLIB)
UNLOAD(TAPE2)
UPDATE(Q,W)
FTN(I=COMPILE,A)
REWIND(PROLIB,LGO)
COPYL(PROLIB,LGO,MYJOB)
REQUEST TAPE8,VSN=12000.  RING IN
MYJOB.
EXIT(S)
7/8/9
```

```
    UPDATE CORRECTION SET
    *COMPILE -------------
7/8/9
    DATA
6/7/8/9
```

COPYL (as explained in the SCOPE manual, page 10-12) selectively replaces the "old" routine on PROLIB with "new" routines from LGO and combines them into file MYJOB which can be executed.

2.8 OTHER ASSOCIATED CAPABILITIES OF UPDATE

2.8.1 Punching Out Source Image Cards

USE: S=PUNCH in the UPDATE CONTROL card

EXAMPLE=UPDATE (Q,W,S=PUNCH)

NOTE: This punches the *DECK AND *COMDECK cards. However, FTN treats these as comments (*) in column 1.

3. UPDATE MANIPULATIONS

UPDATE maintains the source cards in a special compressed form. The usual mode of operations is that it reads this previously created program library, which already exists, from a file called OLDPL. It incorporates any changes indicated and writes card images suitable for passing to a compiler, on a file called COMPILE.

If the N option is chosen, it also makes a file called NEWPL, which is a PL containing both the old cards and the corrections.

UPDATE normally reads cards (directives) from the INPUT file. Some cards are instructions as to what operations to perform. The others are inserted into the PL. Some directives (e.g., *DECK, *COMDECK, and *CALL) go on the PL along with the source cards. They are used by UPDATE to translate from the PL to card images suitable as input to a compiler.

3.1 IDENTIFIERS

When the PL is created, each card on it is assigned a name and sequence number. The name is the same as the deck name and the number is the sequence number within that deck. The *DECK card itself is number one.

A set of changes to the PL (CORRECTION SET) starts with an *IDENT card, and refers to cards in the original deck or in previous correction sets by name and number. The name and number of each card on the compile file is put in columns 73-90, so it appears in the listing from the compiler. Cards added by a CORRECTION SET take their name from the *IDENT card of the CORRECTION SET and their number from their sequence in the CORRECTION SET for that *ID name. CORRECTION SET names, like DECK and COMDECK names, may be from 1 to 9 alphanumeric characters -- no duplication is allowed. Any number of CORRECTION SETS may be used, either on the old PL or in the current job. There is no restriction on the order in which changes are made except for the fact that one change may interact with another if they operate on the same cards.

3.2 COMDECKS

Blocks of cards may be kept on the PL starting with a *COMDECK card. Then, when the deck is being put on COMPILE, if a *CALL NAME card is encountered, the *CALL card is replaced by all the cards in the named COMDECK. A COMDECK must exist ahead of any calls to it, both in time and in its position on the PL.

3.3 DECKS

UPDATE puts onto COMPILE blocks of cards delineated by *DECK cards. If any part of a deck is changed, that entire deck is put out on COMPILE. This includes changes to the deck or to any COMDECK called by it. A *DECK card has an identifier like any other card in the PL, but *DECK cards are not put out on COMPILE so its identifier is not usually written out. *DECK cards may be added or deleted like any other, but to do this usually means that the deck, identifier, and programs or subroutine names do not agree (see below).

For examples of *DECK and COMDECK usage, see para. 2.1.

3.4 ADDITIONS AND DELETIONS

The control cards used to add and delete are *INSERT, *BEFORE, and *DELETE. They may be abbreviated as *I, *B, and *D.

The following correction set shows how to insert, replace, and delete cards in decks that are already in the PL.

```
*IDENT CHANGE
*I SUB1.3
   CARD 1
   CARD 2
*D SUB1.12
   CARD 3
   CARD 4
*D SUB1.15,SUB2.20
   CARD 5
*D SUB2.23
*B SUB1.7
   CARD 6
```

This CORRECTION SET inserts cards 1 and 2 after the card identified as SUB1.3.

Cards 1 and 2 are now known to UPDATE as CHANGE.1 and CHANGE.2.

It deletes SUB1.12 from the active cards, and puts cards 3 and 4 in its place. It deletes all cards from SUB1.15 to SUB2.20, inclusive, including those which have names other than SUB1. and it puts card 5 in their place. It deletes SUB2.23, and it puts card 6 in before SUB1.7.

Cards immediately following a *D card which are not recognized by UPDATE as directives are inserted into the PL after the deleted cards.

3.5 *COMPILE

The card *COMPILE DECKNAME1, DECKNAME2, DECKNAME3, in the correction deck will cause the named decks to be put out on the compile file even if they are not changed by the CORRECTION SET. Any number of decks may be named. The list may not extend beyond column 72. Any number of *COMPILE cards may be present. A *COMPILE card may not come within a CORRECTION SET, that is, it must come just before an *IDENT card or at the end of all the CORRECTION SETS.

3.6 ADDING AN ENTIRE SUBROUTINE

On the original PL, subroutine SUB2 follows subroutine SUB1. The following cards show how a new subroutine, SUB3, could be inserted between SUB1 and SUB2.

```
*ADDFILE INPUT,SUB1
*DECK SUB3
   SUBROUTINE SUB3 (.....ETC.)
   ..............
   END
```

The cards of SUB3 will take their identifying name from the *DECK card, i.e., they will be known to UPDATE as SUB3...... The second argument on the *ADDFILE card could be omitted. SUB3 would then be added onto the end of the PL. Remember the OLDPL is never changed -- the ADDFILE deck appears on file NEWPL.

4. RECAPITULATION

The UPDATE directives used in this writeup, together with their abbreviations and examples are:

DIRECTIVE	ABBREV	PARAMETER(S)	EXAMPLE(S)
*COMDECK	*CD	Unique deck name	*COMDECK DATASET
*DECK	*DK	Unique deck name	*DECK MAINJOB
*CALL	*CL	Name of existing COMDECK	*CALL DATASET
*IDENT	*ID	Unique CORRECTION SET name	*IDENT DEC2074
*INSERT	*I	Single card identifier	*INSERT MERCATR.3
*BEFORE	*B	Single card ID	*BEFORE DEC1873.7
*DELETE	*D	Card ID or 1st card ID, last ID	*DELETE DEC1873.6 *DELETE DEC1873.6, UPDRAFT.2 (1)
*COMPILE	*C	Deck name or name 1, name 2, etc.	*COMPILE MAINJOB *COMPILE MAINJOB, MERCATR
*ADDFILE	*AF	Filename, deck name	*ADDFILE INPUT, SMOOTHER or *ADDFILE INPUT

(1)Note, this directive deletes all cards between DEC1873.6 and UPDRAFT.2 as well as DEC1873.6 and UPDRAFT.2.

Note that, with the exception of the COMDECK example, a series of 3 tapes were used for UPDATE. Initially the creation run produced a Plon VSN 74321. This was then used as the OLDPL to produce VSN 74322 (ADDFILE job). VSN 74322 in turn was used to produce the re-sequenced version on VSN 74323. VSN 74323 was used to generate the UPDATE/OBJECT code tape (VSN 74321) for COPYL & UPDATE(Q) examples. Thus three generations of the PL were available at all times. Using such a system will allow the programmer to "recover" in the event of unforeseen difficulties.

This programming note is presented as a guide to some of the applications and uses of UPDATE. Further explanation and more options are explained in the UPDATE REFERENCE MANUAL.